Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Сетевые информационные технологии и сервисы (СИТиС)»

**ОТЧЁТ**

**о прохождении учебной (технологической) практики**

**на кафедре Сетевых информационных технологий и сервисов (СИТиС)**

Выполнил: студент группы БСТ2001

Коцич Л.

Вариант №9

Проверила: спец. учебн. лаб. каф. СИТиС.

Тришина С.В.

Москва 2022

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**по учебной (технологической) практике**

Создать модель микроконтроллерной схемы на основе Arduino с реализацией точного определения местоположения объекта на ограниченной плоскости. Разработать программу, отображающую работоспособность схемы с возможностью использования базы данных и всей присущей ей информации. Конечный результат проделанной работы отобразить с помощью отчёта и презентации.

**Цель учебной (технологической) практики**:

Главной целью работы является выполнение поставленной задачи, изучение и получение следующих навыков, а именно: построения схем Arduino, работа с симуляторами схем, разработки программного обеспечения для данных схем и построения систем управления базы данных, построения связей предметной области и разработки пользовательского интерфейса. Также обретение умения правильно и грамотно отобразить полученные результаты работы с помощью отчёта и презентации, содержащей в себе видеоматериал.

Для освоения и достижения цели учебной (технологической) практики необходимым является исследование ранее поставленной темы индивидуального задания и выполнение всех поставленных задач, представленных ниже.

**Задачи учебной (технологической) практики:**

1. Составить ER-диаграммы (схемы взаимосвязей сущностей и их атрибутов) предметной области;
2. Определить составные части системы, а также их дальнейшая взаимоувязка в единый макет;
3. Изучить и инсталлировать симуляторы для аппаратно-программных средств семейства Arduino;
4. Составить схему взаимодействия в выбранном симуляторе;
5. Спроектировать базу данных (БД) для предметной области;
6. Спроектировать и реализовать графический интерфейс с самостоятельным выбором языка реализации или фреймворка, а также обеспечить взаимодействие графического интерфейса с БД;
7. Составить отчет и презентации по теме практики.

**Содержание**

1. Работа с предметной областью4

1.1 Описание предметной области4

1.2 Выделение сущностей предметной области и атрибутов сущностей4

1.3 Составление ER-диаграммы5

1.4 Определение составных частей системы5

1. Работа с Arduino7

2.1 Изучение и инсталляция симуляторов Arduino8

2.2 Выбор электронных компонентов9

2.3 Составление схемы взаимодействия в выбранном симуляторе11

2.4 Сборка схемы в реальной жизни12

2.5 Теоретическая часть работы схемы15

1. Проектирование программного обеспечения19

3.1 Проектирование и реализация графического интерфейса20

3.2 Проектирование базы данных24

Заключение26

Список использованных источников27

Приложение А30

Приложение Б34

Приложение В41

Приложение Г45

1. **Работа с предметной областью**

В данном разделе ведется описание предметной области, разбираются имеющиеся в ней сущности и атрибуты сущностей, демонстрируется ER-диаграмма в соответствии с проектом, а также определяются составные части системы, представленные в виде чертежа.

* 1. **Описание предметной области**

Под предметной областью определяют создание системы, определяющей точное местоположение объекта на ограниченной плоскости. Для достижения поставленных задач и выполнения работы требуется сконструировать устройство, представляющее из себя собранные воедино электронные приборы, охватывающие определенную плоскость для измерения местоположения объекта на ней, а также с собственной базой данных для хранения информации, которая будет просматриваться в созданном для проекта графическом интерфейсе.

* 1. **Выделение сущностей предметной области и атрибутов сущностей**

Перед тем как приступить к созданию самого проекта и производить последующую работу с ним, следует для понимания определить необходимые сущности предметной области и атрибуты сущностей. На основе предметной области можно выделить следующее:

Плоскость: размер плоскости;

Объект: координаты;

Электроприборы: ультразвуковой датчик A, ультразвуковой датчик B, плата Arduino Uno R3.

Таким образом, данные сущности и атрибуты будут использоваться в работе, а также более подробно будут продемонстрированы на ER-диаграмме.

* 1. **Составление ER-диаграммы**

Следует собрать все имеющиеся сущности и атрибуты в виде удобной для понимания ER-диаграмме, представленной на рисунке 1.3.1.

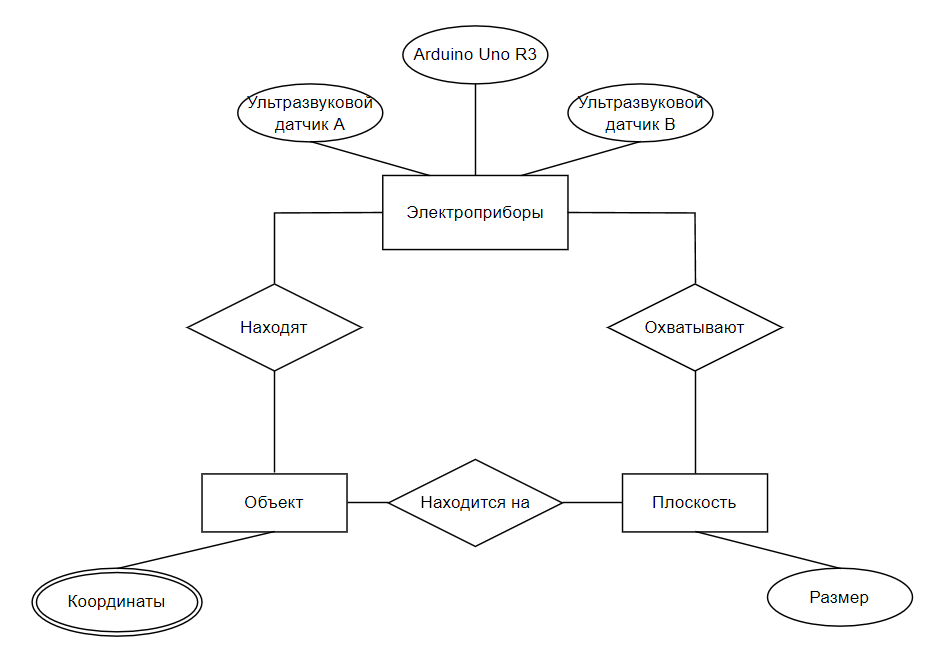


Рисунок 1.3.1 – ER-диаграмма

Данная схема была выполнена в нотации Чена так как она является удобочитаемой и эффективной в плане использования пространства для рисования. Множества сущностей в ней изображаются в виде прямоугольников, а их отношения изображаются в виде ромбов. Стоит отметить, что когда сущность участвует в отношении, то они связаны линией, а в случае, если связь необязательно, то ее выделяют пунктирной линией. Атрибуты отображаются в виде овалов и связываются линией с одним отношением, либо с одной сущностью.

* 1. **Определение составных частей системы**

Каждая разработка требует чертеж для понимания того, как именно ее собирать в дальнейшем. Таким образом после определения предметной области, сущностей и атрибутов, и составления ER-диаграммы, был разработан чертеж, который был взаимоувязан в единый макет, представленный на рисунке 1.4.1.

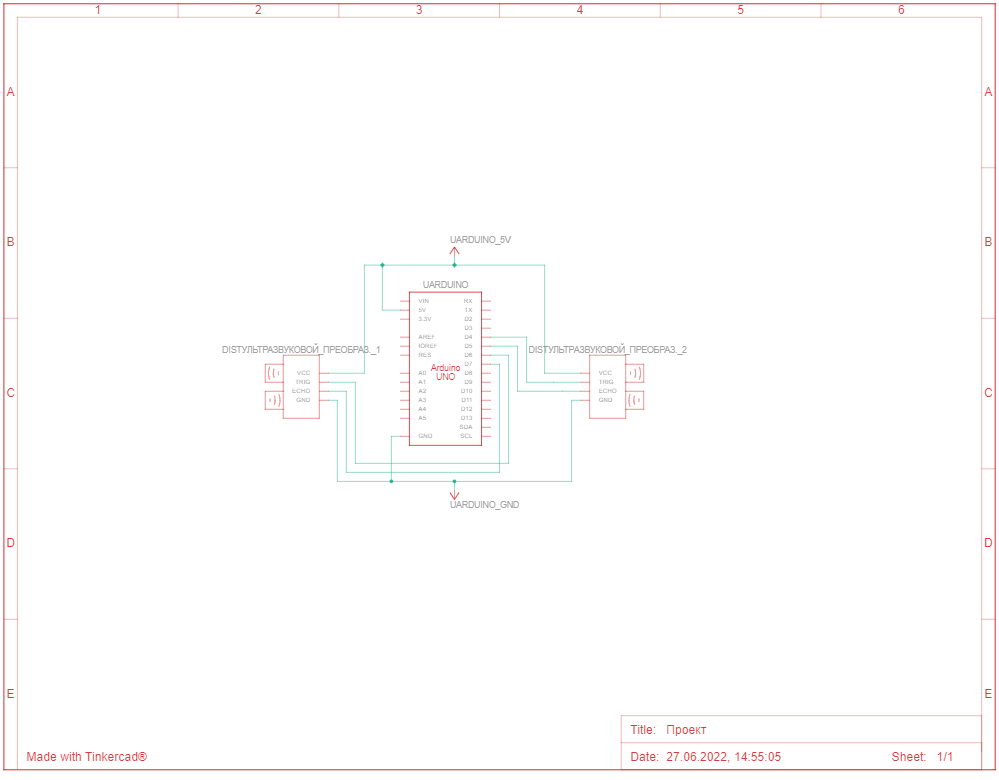


Рисунок 1.4.1 – Макет схемы

1. **Работа с Arduino**

Данный раздел является одним из важнейших частей работы, так как в нем описаны все пункты, связанные с работой Arduino, а именно: взаимодействие как и с симулятором Arduino, так и с его физической оболочкой, то есть с сами электронным устройством, выбор электронных компонентов, необходимых для выполнения работы и для функционирования макета, а также составление всех схем взаимодействия. Также в разделе представлены формулы, благодаря которым происходит работа компонентов и схемы конечных конструкций.

Говоря об Arduino[6], можно сказать что это инструмент для проектирования электронных устройств, либо же это электронный конструктор, более плотно взаимодействующий с окружающей физической средой, чем стандартные персональные компьютеры, которые фактически не выходят за рамки виртуальности. Применяется для создания электронных устройств с возможностью приема сигналов от различных цифровых и аналоговых датчиков, которые могут быть подключены к нему, и управления различными исполнительными устройствами. Проекты устройств, основанные на Arduino, могут работать самостоятельно или взаимодействовать с программным обеспечением на компьютере.

Для сборки виртуальной схемы был выбран симулятор Tinkercad. Tinkercad[14] – это бесплатное кроссплатформенное веб-приложение для 3D-проектирования, работы с электронными устройствами и написания программного кода. Tinkercad очень прост в освоении и имеет в себе обширное количество электронных устройств, которые необходимы для реализации схемы. Удобством эмулятора является возможность запуска моделирования схемы для того, чтобы примерно понять и оценить работоспособность самой схемы и кода, внесенного в Arduino.

Также были использованы такие среды разработки, как Arduino IDE, Processing IDE и Visual Studio 2019 IDE, описание и принцип работы которых представлен в последующих подразделах. Языки, которые были применены для реализации всего проекта – это C++, C# и Java.

* 1. **Изучение и инсталляция симуляторов Arduino**

Первым делом были рассмотрены все предложенные симуляторы, среди которых самыми популярными являются: SimulIDE, WOKWI и Tinkercad. Наилучшим вариантом как отмечалось ранее является Tinkercad, так как в сравнении с другими симуляторами, данный симулятор имеет большее количество электронных устройств, необходимых для реализации макета. На рисунке 2.1.1 для ознакомления с Tinkercad представлен пустой проект, в котором в дальнейшем будет собрана схема из имеющихся электронных устройств.

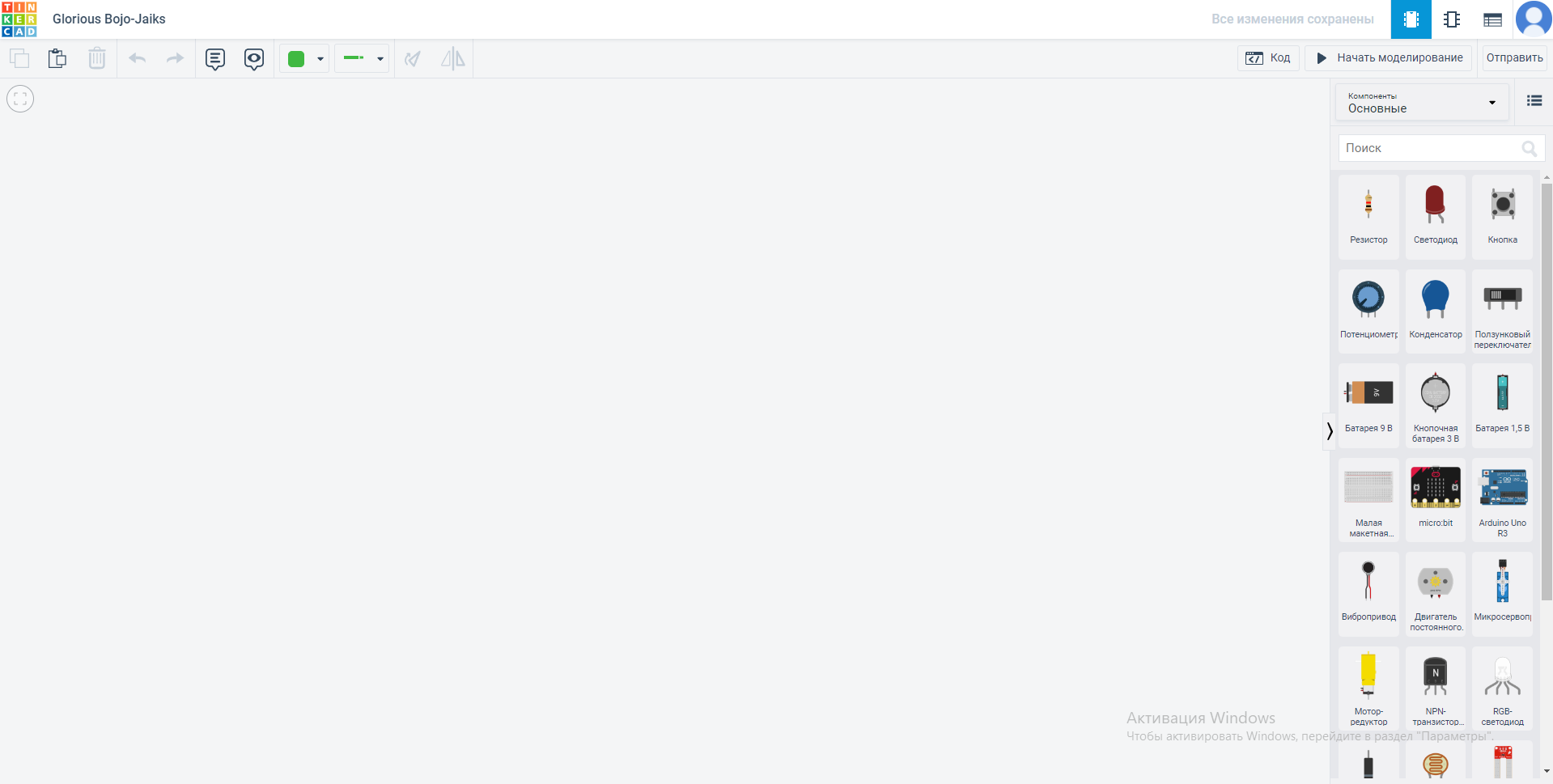


Рисунок 2.1.1 – Tinkercad

Как видно из рисунка, представленного выше, Tinkercad имеет достаточно понятный интерфейс, а также большое количество функций, дающих возможность реализовать свой проект наилучшим образом. В правой части экрана перед пользователем находятся элементы, а сам запуск схемы будет осуществлен с помощью кнопки «Начать моделирование».

* 1. **Выбор электронных компонентов**

Основной упор при выборе электронных компонентов уходит в совместимости компонентов, в их характеристики, соответствующие поставленной для выполнения задаче, а также в стоимость. Вся стоимость представленная ниже приведена и рассчитана за июнь месяц 2022 года, а сами устройства куплены в ТК «Митинский Радиорынок», так как именно там располагаются достаточно дешевые экземпляры в большом количестве.

Самый первый и самый важный электронный компонент – это Arduino Uno R3. Arduino Uno R3[16] – это плата микроконтроллера на базе ATmega328P с рабочим напряжением в 5 В, имеющая 14 цифровых контактов ввода/вывода, 6 аналоговых входов, керамический резонатор 16 МГц, USB-соединение, разъем питания, коннектор ICSP и кнопку сброса. Он имеет все необходимое для поддержки микроконтроллера и благодаря USB-кабелю типа USB в USB-B можно загружать и перезаписывать программы. Стоимость Arduino Uno R3 составляет 900 рублей, а стоимость USB-кабеля составляет 100 рублей.

Вторым компонентом является ультразвуковой датчик расстояния HY-SRF05[9]. Для реализации проекта потребовалось два таких датчика, причиной для этого послужила необходимость нахождения местоположения объекта на плоскости, а именно получение координаты X и координаты Y с помощью ультразвуковых датчиков. Принцип нахождения описан в дальнейшей части работы. Сам модуль имеет аналог HC-SR04, который можно взять в случае отсутствия HY-SRF05 так как можно не изменять программный код и так как их разница заключается в том, что модуль HY-SRF05 имеет более высокое разрешение и специальный выход OUT, который переводит датчик в режим работы по одному проводу, таким образом можно сократить количество используемых выводов микроконтроллера. Также HY-SRF05 выбран по той причине, что его точность имеет лучший показатель по сравнению с HC-SR04, а именно 2 мм вместно 3 мм. Принцип работы датчика заключается в том, что модуль формирует и посылает частотную посылку, затем переключается в режим приема и по задержке отраженного сигнала формирует импульс шириной, пропорциональной расстоянию до объекта. Ультразвуковой дальномер HY-SRF05 обеспечивает бесконтактное дистанционное измерение расстояния от 2 см до 4.5 метров, с точностью измерения до 2 мм. Разъём сенсора HY-SRF05 представляет из себя пять обычных штырьков пинов (тип папа) с шагом 2.54 мм. Напряжение питания модуля составляет 5 В постоянного тока. Стоимость одного такого датчика составляет 150 рублей, тем самым в конечном счете понадобилось 300 рублей.

Также требуются 4 изолированных провода разного цвета длинной в 2,5 метра и 1 провод монтажный медный гибкий с ПВХ изоляцией длинною в 76 сантиметров. 4 длинных провода связывает Arduino Uno R3 с 2 ультразвуковыми датчиками HY-SRF05 путем пайки, а 1 провод монтажный используется в качестве каркаса для того, чтобы устойчиво держать датчики на определенной высоте. Стоимость суммарная составила 188 рублей, где стоимость монтажного провода составляет 38 рублей, а 4 длинных провода составляют 150 рублей. Принцип работы ультразвукового датчика представлен ниже на рисунке 2.2.1:

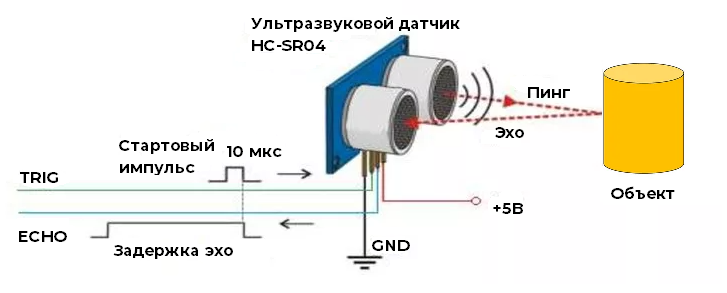


Рисунок 2.2.1 – Принцип работы ультразвукового датчика

Для того, чтобы присоединить провода к датчикам потребовались контакт гнездо для дюпонт разъема размеров 2,54 мм в количестве 8 штук. Стоимость составила 84 рубля.

Необходимыми для пайки являются следующие предметы: канифоль стоимостью 122 рубля, флюс стоимостью 55 рублей, кислота стоимостью 98 рублей, припой оловянный стоимостью 121 рубль, плоскогубцы 176 рублей и сам паяльник стоимостью 157 рублей. Таким образов на данный паяльный набор потребовалось 729 рублей.

Для крепления датчиков к ножкам, сделанных из провода, а также разграничения проводов для повышения безопасности и защиты от замыкания или возгорания потребуется изолента синего цвета, которая стоит 56 рублей.

Таким образом на реализацию всего макета понадобилось 2357 рублей, а если не включать в стоимость паяльный набор, то выходит 1628 рублей, что является в обоих случаях самым дешевым из возможных вариантов сборки.

* 1. **Составление схемы взаимодействия в выбранном симуляторе**

Требуется составить схему в симуляторе Tinkercad используя ранее перечисленные компоненты. Данный шаг поможет понять как именно в дальнейшем следует собрать схему вне симулятора, а также плюсом является возможность безопасно проверить ее работоспособность без каких-либо последствий. Сама схема представлена на рисунке 2.3.1.

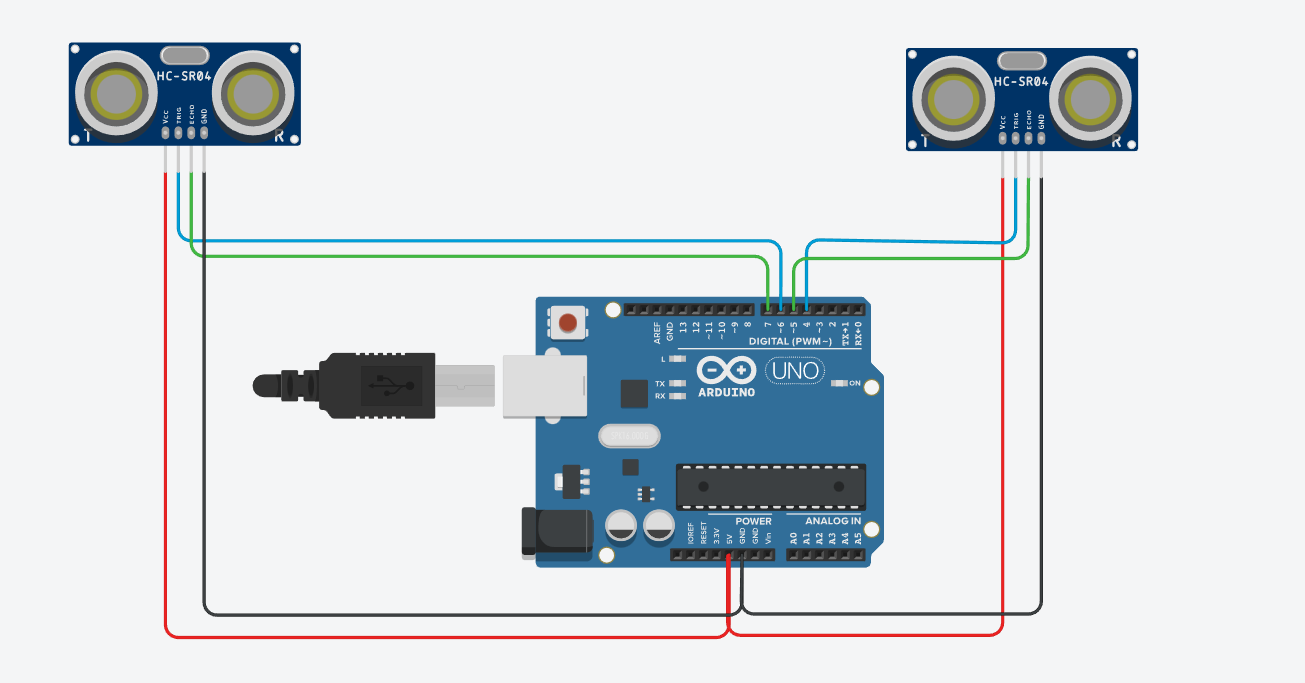


Рисунок 2.3.1 – Собранная схема в Tinkercad-е

Рассматривая рисунок 2.3.1, следует отметить, что в симуляторе используется другой ультразвуковой датчик, являющийся идентичным HY-SRF05, то есть HC-SR04. Оба датчика подключены верно и в соответствии с их выводами. Сами выводы можно более подробно описать следующим образом:

1. Вывод с меткой «VCC» – плюс питания (+5 В);
2. Вывод с меткой «TRIG» – вход сигнала триггерного управления;
3. Вывод с меткой «ECHO» – выход эхо-сигнала;
4. Вывод с меткой «GND» – цифровой выход есть/нет.

Но также следует отметить, что HY-SRF05 имеет вывод с меткой «OUT» – цифровой выход есть/нет.

Для соединения были использованы провода 4 разных цветов для лучшего понимания того, какой именно провод чему соответствует. Аналогичным образом и цветами собрана схема в реальной жизни.

* 1. **Сборка схемы в реальной жизни**

После того, как схема была собрана в симуляторе и проверена на правильность, следующим шагом является сборка схемы в реальной жизни. Первым делом следует теоретически представить то, как именно должен работать весь проект. Сама задумка и вся теория изначально были расписаны в бумажном варианте, так как были внесены поправки, велись расчеты и рисовалось для понимания от руки (см. Приложение Г), но ниже представлены рисунки, переведенные в цифровой формат и нарисованные в таком веб-приложении как Figma. Говоря о Figma[20] можно сказать что это онлайн-сервис, предназначенный для создания прототипов сайтов или приложений, иллюстраций и векторной графики. В данном же случае требуется создать иллюстрации, демонстрирующие как именно работает проект, область обнаружения объекта устройствами, необходимые геометрические расчеты, а также саму схему установки.

Так как имеются два ультразвуковых датчика, то следует отобразить какую часть они покрывают и выделить то пространство, являющееся ограниченной плоскостью для дальнейшего нахождения местоположения объекта на нем. Область обнаружения объекта представлена на рисунке 2.4.1.

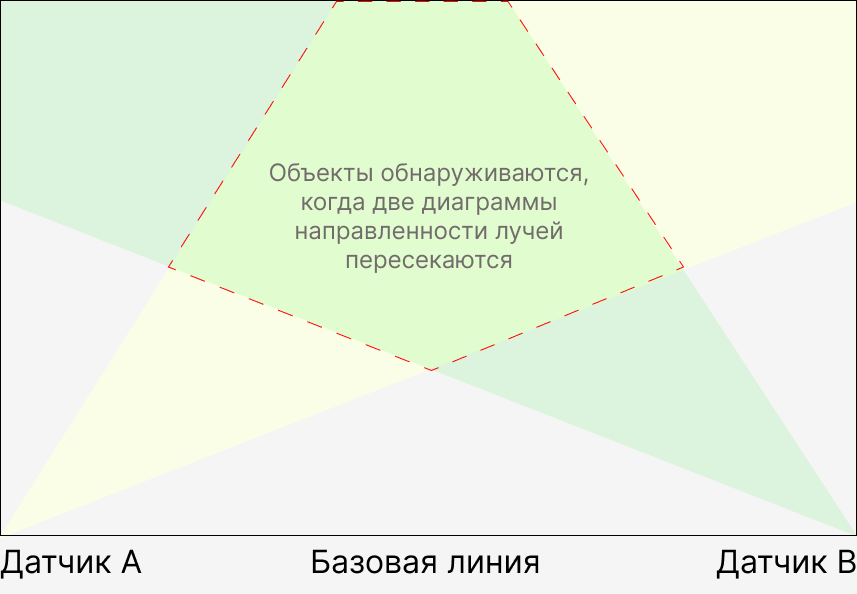


Рисунок 2.4.1 – Собранная схема в Tinkercad-е

Рассматривая рисунок 2.4.1 видно, что на нем изображены две наложенные диаграммы направленности для датчика (преобразователя) A и B. Ультразвуковой датчик с левой стороны после отправления звукового сигнала принимает эхо от любого объекта в зоне с желтым цветом, в то время как второй датчик с правой стороны будет получать эхо лишь в тех случаях, когда объект располагается в зеленой зоне. Если объект не находится в данной зоне, то в таком случае невозможным является определение координат объекта и в последующем местоположения объекта на данной ограниченной плоскости, указанной в середине рисунка 2.4.1. Чем шире расположить датчики, тем увеличивается возможность обнаружения в зоне.

Из рисунка представленного выше становится понятным что один из датчиков не должен иметь часть, посылающую звуковой сигнал, так как требуется принимать только то эхо, отправленное от первого датчика и полученное от отражения от объекта. В случае если запускать в симуляторе оба модуля, то Arduino пытается найти два объекта и нахождение объекта становится проблематичным и менее точным, поэтому было принято решение заклеить изолентой в несколько слоев и накрыть какой-либо крышечкой от любой бутылки передающий (Т) преобразователь для того, чтобы заблокировать распространение ультразвука тем самым делая датчик пассивным. Именно по этой причине следует отобразить две схемы сборки датчика, так как первая конструкция будет собрана в обычном варианте, а вторая будет с закрытой частью передающего (Т) преобразователя. На рисунке 2.4.2 изображена схема датчика A:

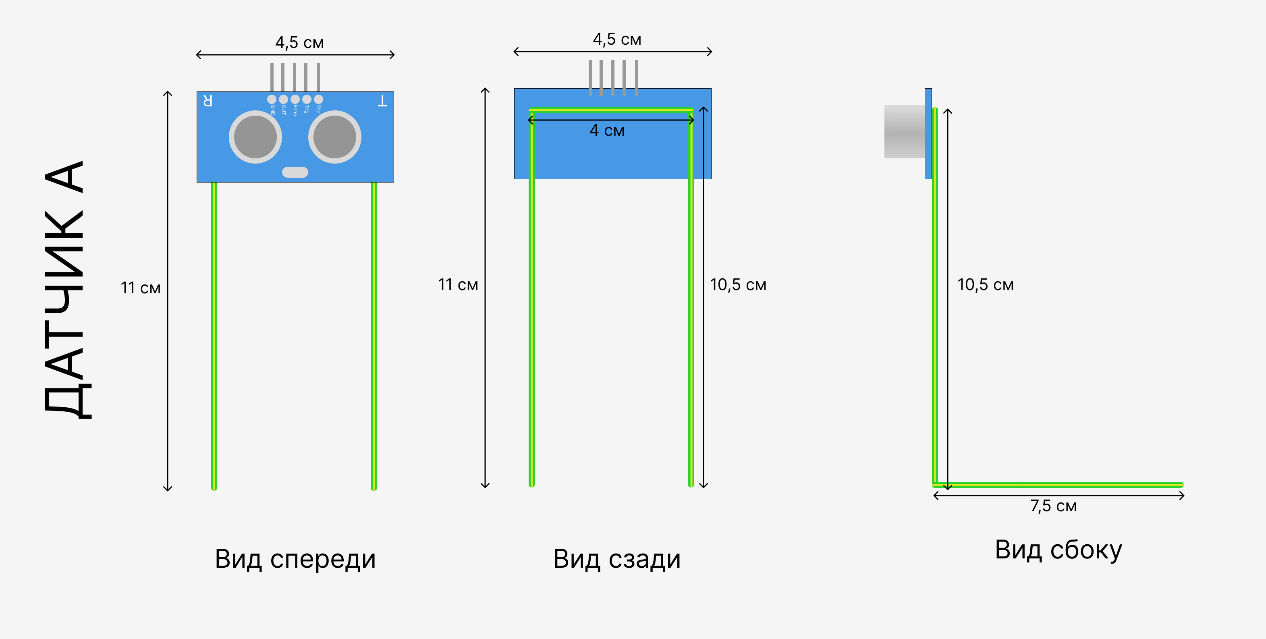


Рисунок 2.4.2 – Схема датчика A

Рассматривая датчик B, можно сказать, что он полностью идентичный датчику А, лишь требуется закрыть ранее указанную часть на ультразвуковом датчике. Результат представлен на рисунке 2.4.3:

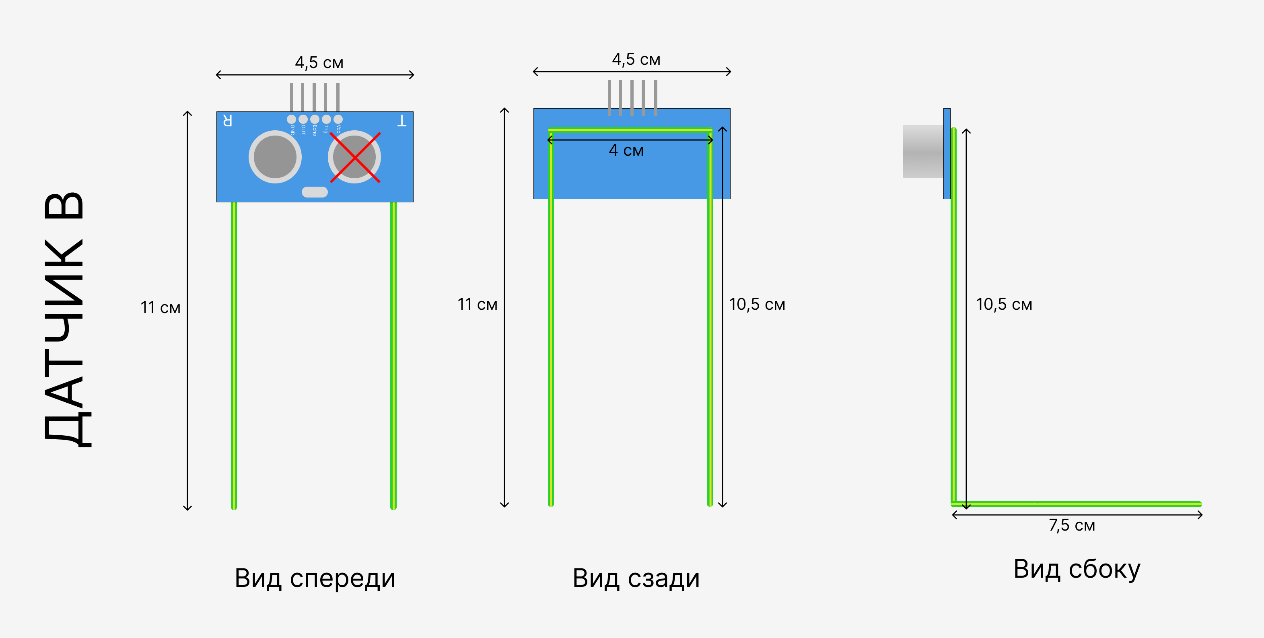


Рисунок 2.4.3 – Схема датчика B

* 1. **Теоретическая часть работы схемы**

После сборки всей конструкции следует разобраться в том, как работает сама схема. Вся схема состоит из Arduino, двух ультразвуковых датчиков и формулы Герона, используемая для треугольников. Благодаря данной формуле возможен расчет площади любого из возможных треугольников при известных его сторонах. После того, как площадь станет известной, требуется произвести расчет местоположения объекта на самой плоскости, которая находится относительно известной базовой линии, используя знания в области тригонометрии и теорему Пифагора. Для проведения расчетов следует изобразить небольшую схему треугольника и подписать определенные элементы, которые в дальнейшем понадобятся для формул. Схема представлена на рисунке 2.5.1.

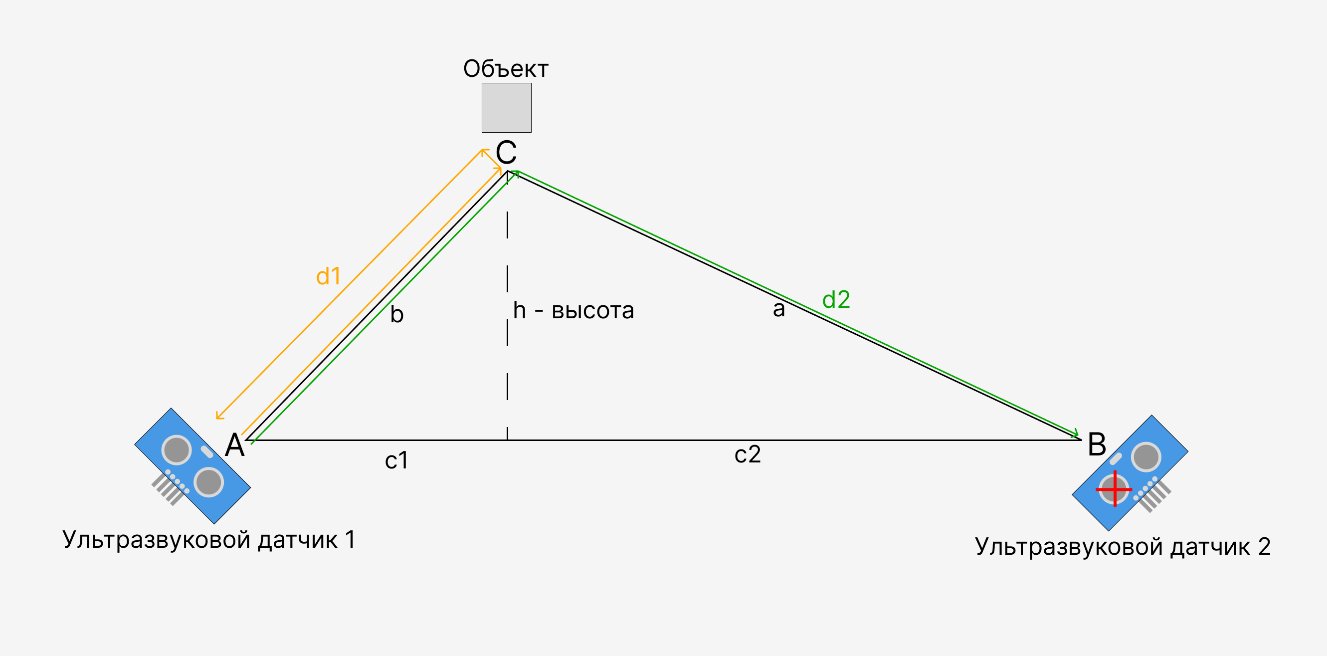


Рисунок 2.5.1 – Схема датчика B

Как известно площадь любого треугольника может быть рассчитана по формуле, в которой произведение основания и высоты делится на 2. Из этого следует, что если требуется найти координату Y, то есть высоту, то следует умножить значение площади на 2 и полученное значение разделить на основание. Сама суть заключается в том, чтобы поставить датчики в таком положении, чтобы расстояние между ними было фиксированным (базовая линия), а благодаря самим датчикам произвести замер расстояния от датчиков до объекта с использованием ультразвука. Сам процесс представлен на рисунке 2.5.1 выше.

Из рисунка видно, что первый ультразвуковой датчик посылает импульс, который в дальнейшем отражается от объекта во всех направлениях, а сам импульс будут принимать оба датчика, как и первый, так и второй. Второй ультразвуковой датчик не посылает никаких импульсом, он исключительно принимает их. После того, как импульс был послан, то он преодолевает обратное расстояние, показанное желтым цветом на рисунке выше. Если разделить путь на два и учитывать скорость звука, то благодаря данным выводам можно произвести расчет расстояния d1 с помощью формулы:

где t – время в микросекундах.

59 является константным значением и получается оно следующим образом:

1. Скорость звука составляет 340 м/с, а это 0,034 см/мкс (сантиметр/микросекунда);
2. Обратное значение 0,034 см/мкс составляет 29,412 мкс/см, которое при умножении на 2, дает в результате 58,824 или же 59 в случае округления.

Умножается значение на 2 для того, чтобы получить длину обратного пути.

Второй путь, представленный зеленым цветом, исходит из первого ультразвукового датчика и идет до второго ультразвукового датчика. Если из данного пути вычесть расстояние, которое было первым, то есть d1, то таким образом можно рассчитать расстояние d2. Таким образом мы получаем следующую формулу:

Сама величина 29,5 также является константным значением и получается таким образом, что в данном случае отсутствует обратный путь, поэтому используется 29,5, что представляет из себя половину значения в формуле (1) выше.

Таким образом были получены все длинны сторон треугольника ABC и благодаря формуле Герона можно рассчитать и найти координату Y. Сама формула Герона использует полупериметр, в котором добавляется каждая из сторон треугольника и полученный результат делится на два. Нагляднее можно выразить следующей формулой:

Теперь же площадь можно рассчитать по формуле, представленной ниже:

В конечном счете после того, как будет известна площадь, то будет возможным рассчитать ранее упомянутую высоту, то есть координату Y.

Переходя к координате X, следует уточнить, что для того, чтобы узнать координату, требуется воспользоваться теоремой Пифагора. Вычисление происходит таким образом, что, откладывая из вершины треугольника перпендикулярную линию до базовой линии, можно получить прямоугольный треугольник. Таким образом координата X получается из формулы Пифагора, представленной ниже:

Тем самым после сборки всей схемы и разбора теории становится намного понятнее то, как написать код, который в дальнейшем будет реализовывать всю систему.

1. **Проектирование программного обеспечения**

Последним и самым ключевым этапом работы является проектирование программного обеспечения, благодаря которому все раннее описанные пункты будут взаимодействовать воедино, и работа будет реализована в точности с ее поставленной задачей. Как было уже отмечено для проектирования программного обеспечения были использованы такие программы как Arduino IDE, Processing IDE и Visual Studio 2019 IDE. В зависимости от среды разработки были использованы соответствующие им языки программирования. Таким образом для написания кода для платы Arduino потребовался C++, для Processing потребовалась Java, а для конечного создания графического интерфейса, который будет взаимодействовать со всеми остальными частями работы потребовался C#.

Говоря о самих программах, можно сказать следующее:

Arduino IDE[21] – это приложение, которое позволяет составлять программы в удобном текстовом редакторе, компилировать их в машинный код, и загружать на все версии Arduino.

Processing IDE[10] – это графическая библиотека и интегрированная среда разработки (IDE), созданная для сообществ в области электронного искусства, нового медиа-искусства и визуального дизайна с целью обучения непрограммистов основам компьютерного программирования в визуальном контексте. Обработка использует язык Java с дополнительными упрощениями, такими как дополнительные классы и математические функции, и операции с псевдонимами.

Visual Studio 2019 IDE[18] - это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Все программы являются полностью бесплатными и легкодоступными для скачивания из интернета, после установки следует перейти к реализации программной части проекта.

* 1. **Проектирование и реализация графического интерфейса**

Перед тем как соединить все в единый графический интерфейс требуется по порядку спроектировать и реализовать каждую часть работы программного вида. Первым на очереди является разработка кода для Arduino (см. Приложение А), который в дальнейшем будет загружен на саму плату с использованием USB. Для наглядности ниже на рисунке 3.1.1 представлена сама программа и написанный в ней код:

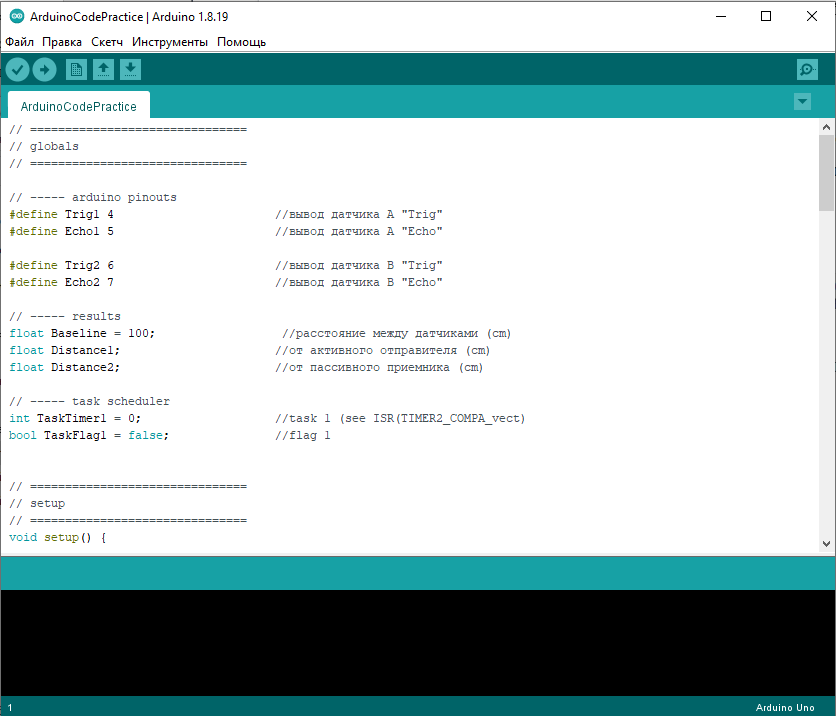


Рисунок 3.1.1 – Программа в Arduino IDE

Для понимания того, что происходит в коде было решено оставлять комментарии у каждой значимой строчки. Изначально комментарии давались на английском языке, но в последствии определенные комментарии для понимания были переведены. Таким образом, когда весь код написан, то следующим шагом является загрузка программы на плату Arduino. Первым делом в разделе «Инструменты» выбирается плата «Arduino Uno» и указывается порт, к которому подключена Arduino. После того, как было произведено подключение, следует сперва нажать кнопку «Проверить» для того, чтобы перепроверить правильность написанного кода и уже за этим следует нажать «Загрузить» для загрузки кода на плату.

Следующим будет Processing IDE, в котором необходимо реализовать все формулы, представленные в теоретической части ранее, чтобы в дальнейшем в Processing-е сделать графический интерфейс с отображением местонахождения объекта на плоскости. Все необходимые и важные строчки прокомментированы в коде для общего понимания (см. Приложение Б). Сам Processing при запуске принимает через код значение порта и использует подключенную к компьютеру Arduino. Благодаря датчикам и параметрам в коде производится весь основной расчет и само нахождение объекта. Ниже на рисунке 3.1.2 представлена сама программа Processing и для наглядности представлен написанный код:

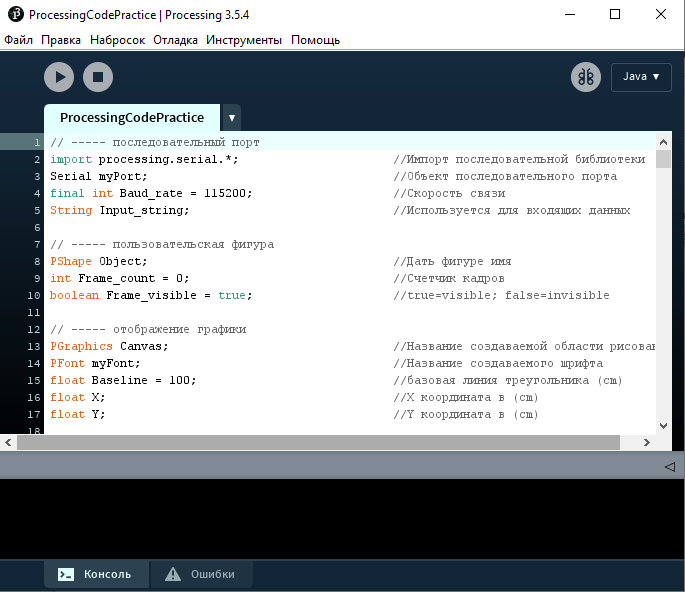


Рисунок 3.1.2 – Программа в Processing IDE

Следует отметить, что следует выбрать свой COM-порт, к которому подключена Arduino. Для этого достаточно посмотреть в программе Arduino IDE к каком порту подключена плата, либо же запустить программный код в Processing-е и посмотреть в высветившемся списке доступные COM-порты. В случае если COM-порт был правильно, тогда появится плоскость, а на ней красная точка. На рисунке 3.1.3 красной стрелкой указано, где именно требуется изменять значение, соответствующее COM-порту.

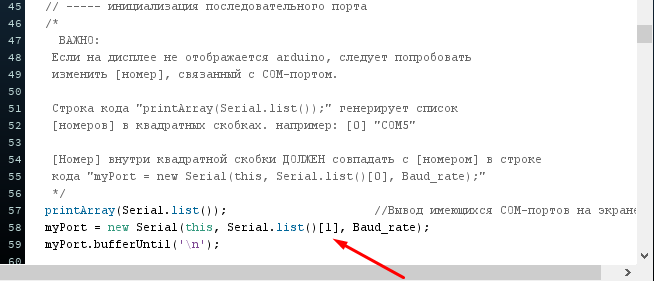


Рисунок 3.1.3 – Выбор COM-порта

Ниже на рисунке 3.1.4 представлена сама плоскость, а также красная точка, обозначающая сам объект. Благодаря такой плоскости становится ясным, где именно находится объект, а в сам Processing выводятся все значения переменных и результаты, рассчитанные по формулам.

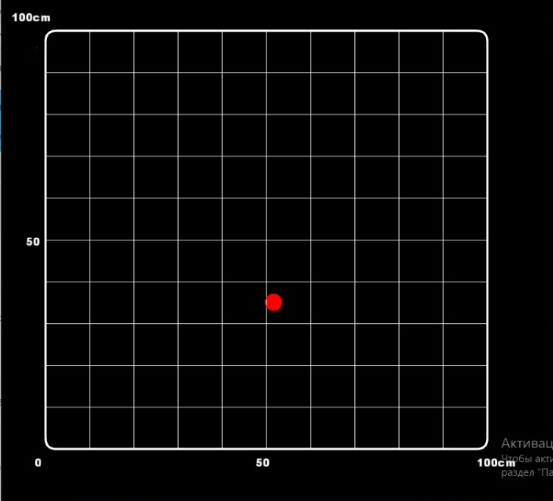


Рисунок 3.1.4 – Запуск программы в Processing IDE

Последней программой, в которой выполнен графический интерфейс с возможностью взаимодействия со всеми предыдущими программами, а также возможностью запуска базы данных, является Visual Studio 2019 IDE. Изначально было решено, что код будет написан на таком языке программировании, как C# с использованием Windows Forms (см. Приложение В). Сам вид программы, а именно разработанный графический интерфейс представлен на рисунке 3.1.5. Дополнительно также была установлена библиотека «MaterialSkin», дающая возможность гибко и красиво разработать сам дизайн приложения. Были использованы кнопки «MaterialButton» для того, чтобы открывать с их помощью программы и файлы.

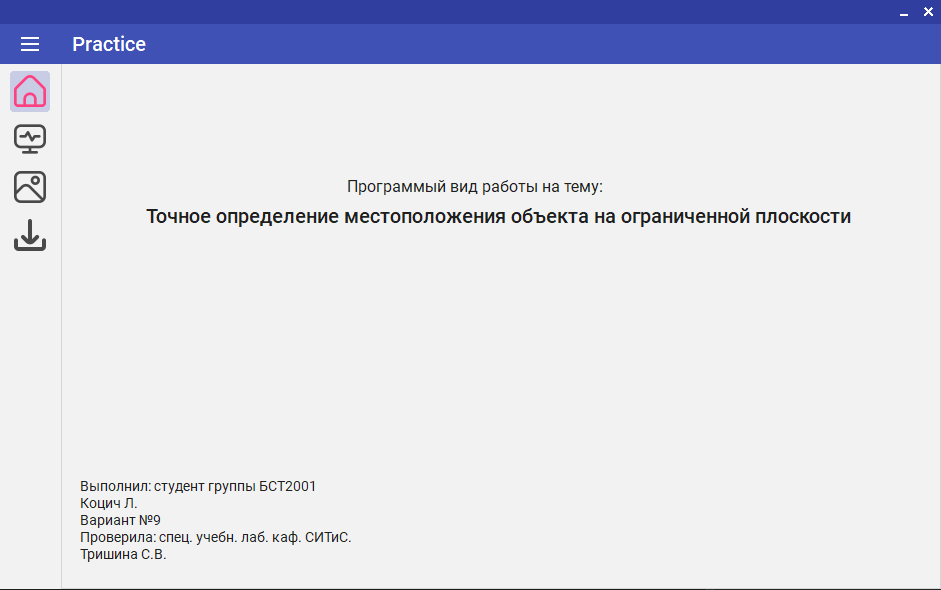


Рисунок 3.1.5 – Графический интерфейс

Сама программа имеет в себе несколько разделов, благодаря которым пользователь может воспользоваться и рассмотреть каждую часть всей работы. С помощью раздела «Start», представленного на рисунке 3.1.6, имеется возможность открыть программы Arduino и Processing, а также открыть базу данных и просмотреть все записи в ней. Интерфейс программы довольно лаконичный и достаточно понятный для освоения любым пользователем. Также в разных разделах имеются дополнительные подсказки, в случае если пользователь хочет побольше узнать о возможностях той или иной кнопки, вызывающей определенные функции.

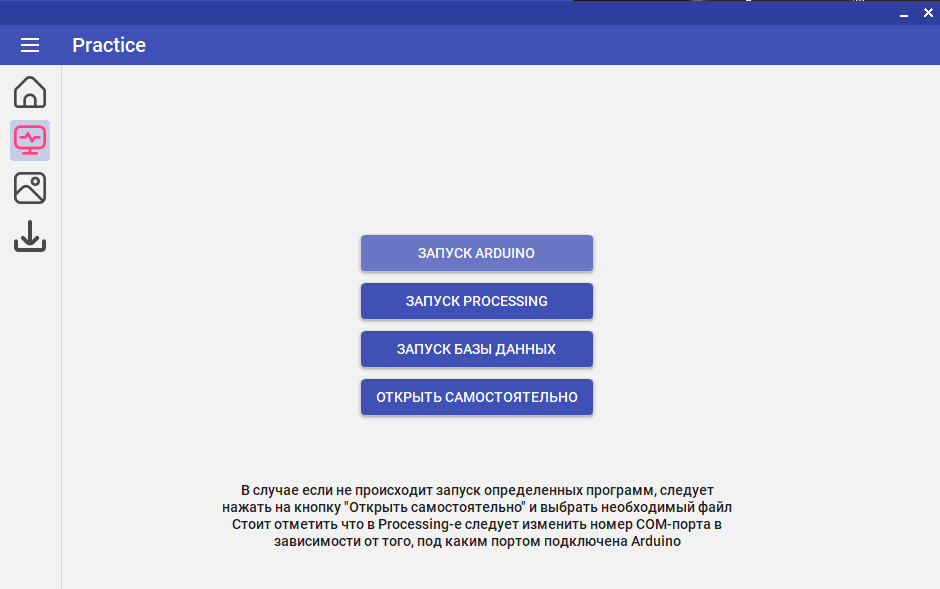


Рисунок 3.1.6 – Раздел «Start»

Также есть такой раздел, как «Pictures» в котором можно ознакомиться со схемой и чертежом проекта, а также дополнительные изображения, которые применялись в течение разработки. Следующий раздел – «Download». В данном разделе можно скачать все файлы, программы и изображения с GitHub, а также скачать саму Arduino IDE и Processing IDE.

* 1. **Проектирование базы данных**

Последним шагом является реализация базы данных. Сама база данных является важной частью в различных проектах и используется для внесения или получения той или иной информации. В данном же случае база данных записывает координаты X и Y, полученные из Processing. Реализована база данных в виде блокнота, представленного на рисунке 3.2.1. Для того, чтобы создавался файл и вносились координаты было необходимо в коде Processing-а дописать некоторые функции, помеченные комментарием «ФАЙЛ» (см. Приложение Б). Сам файл называется «positions» и сохраняется в той же папке, где и программный код Processing. А для того, чтобы воспользоваться самой базой данных достаточно запустить программу в Processing-е и нажать в графическом интерфейсе кнопку «Запуск базы данных», тем самым откроется блокнот со значениями, представленный на рисунке 3.2.2.

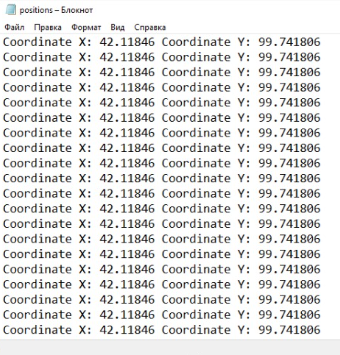


Рисунок 3.2.2 – База данных

**Заключение**

Реализовав полностью поставленное индивидуальное задание, можно сделать вывод о том, что была создана модель микроконтроллерной схемы на основе Arduino с реализацией точного определения местоположения объекта на ограниченной плоскости. Была разработана программа, отображающая работоспособность схемы с возможностью использования базы данных и всей присущей ей информацией.

Отталкиваясь от задания можно сказать, что поставленная цель была достигнута, так как были получены многие навыки, связанные с построением схем Arduino, работой с симуляторами схем, разработкой программного обеспечения для данных схем и построение систем управления базы данных, построением связей предметной области и разработки пользовательского интерфейса. Был разобран теоретический и практический вид работы и представлен в разделах, представленных выше.

Работу можно считать выполненной, так как следующие задачи были реализованы:

1. Составлена ER-диаграммы предметной области;
2. Определены составные части системы, а также произведена их дальнейшая взаимоувязка в единый макет;
3. Изучены и инсталлированы симуляторы для аппаратно-программных средств семейства Arduino;
4. Составлены схемы взаимодействия в выбранном симуляторе и в реальной жизни;
5. Спроектирована база данных;
6. Спроектирован и реализован графический интерфейс, а также обеспечено взаимодействие графического интерфейса с БД;
7. Составлен отчет и презентация по теме практики.

**Список использованных источников**

1. ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) – URL: <https://docs.cntd.ru/document/1200157208> (дата обращения 09.06.2022). – Текст: электронный.
2. ГОСТ 2.001-2013. Межгосударственный стандарт. Единая система конструкторской документации. Общие положения (введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 22 ноября 2013 г. N 1628-ст) – URL: <https://docs.cntd.ru/document/1200106859> (дата обращения 09.06.2022). – Текст: электронный.
3. ГОСТ Р ИСО 12615-2013. Библиографические ссылки и идентификаторы источников для терминологической работы (утв. приказом Федерального агентства по техническому регулированию и метрологии от 8 ноября 2013 г. N 1384-ст) – URL: <https://docs.cntd.ru/document/1200108542> (дата обращения 09.06.2022). – Текст: электронный.
4. ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) – URL: <http://docs.cntd.ru/document/1200157208> (дата обращения: 09.06.2022). – Текст: электронный.
5. ГОСТ 2.105-95. Межгосударственный стандарт. Единая система конструкторской документации. Общие требования к текстовым документам (введен в действие постановлением Госстандарта РФ от 8 августа 1995 г. N 426) – URL: <https://docs.cntd.ru/document/1200001260> (дата обращения: 09.06.2022). – Текст: электронный.
6. Arduino.ru: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://arduino.ru/Reference> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
7. ZAOCHNIK: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://zaochnik.ru/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
8. ВСЁ ДЛЯ ARDUINO В ОДНОМ МЕСТЕ: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://iarduino.kz/blog/sravnenie-ultrazvukovyh-datchikov-hcsr04-or-hysrf05.hml> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
9. Robot-kit: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://robot-kit.ru/3093/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
10. Всемирная Викимедиа: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://www.hmong.press/wiki/Processing_(programming_language)> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
11. Radio Prog: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://radioprog.ru/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
12. Electrikmaster: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://electrikmaster.ru/kak-pravilno-payat/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
13. ТК Митинский Радиорынок: [сайт]. – Москва. – Обновляется в течении суток. – URL: <http://www.tkmitino.ru/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
14. Tinkercad: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://www.tinkercad.com/dashboard> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
15. Справочник языка Processing: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://wikihandbk.com/wiki/Processing:%D0%A1%D0%BF%D1%80%D0%B0%D0%B2%D0%BE%D1%87%D0%BD%D0%B8%D0%BA_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0_Processing/createWriter> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
16. Arduino: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://www.arduino.cc/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
17. Processing: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://processing.org/download> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
18. Visual Studio: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://visualstudio.microsoft.com/ru/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
19. Diagrams: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://online.visual-paradigm.com/diagrams/features/erd-tool/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
20. Figma: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://www.figma.com/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.
21. ROBOTCLASS: [сайт]. – Москва. – Обновляется в течении суток. – URL: <https://robotclass.ru/tutorials/arduino-ide/> (дата обращения 11.06.2022). – Режим доступа: свободный. – Текст. Изображение: электронный.

**Приложение А**

Ниже представлен листинг кода, написанного в Arduino IDE на C++:

Листинг А.1:

// ===============================

// globals

// ===============================

// ----- arduino pinouts

#define Trig1 4 //вывод датчика A "Trig"

#define Echo1 5 //вывод датчика A "Echo"

#define Trig2 6 //вывод датчика B "Trig"

#define Echo2 7 //вывод датчика B "Echo"

// ----- results

float Baseline = 100; //расстояние между датчиками (cm)

float Distance1; //от активного отправителя (cm)

float Distance2; //от пассивного приемника (cm)

// ----- task scheduler

int TaskTimer1 = 0; //task 1 (see ISR(TIMER2\_COMPA\_vect)

bool TaskFlag1 = false; //flag 1

// ===============================

// setup

// ===============================

void setup() {

// ----- configure serial port

Serial.begin(115200);

// ----- configure arduino pinouts

pinMode(Echo1, INPUT); //сделать входы эхо-выводов

pinMode(Echo2, INPUT);

pinMode(Trig1, OUTPUT); //сделать ВЫВОД триггерных выводов

pinMode(Trig2, OUTPUT);

digitalWrite(Trig1, LOW); //установите триггерные контакты на LOW

digitalWrite(Trig2, LOW);

// ----- configure Timer 2 to generate a compare-match interrupt every 1mS

noInterrupts(); //отключение прерываний

TCCR2A = 0; //очистить регистры управления

TCCR2B = 0;

TCCR2B |= (1 << CS22) | //16MHz/128=8uS

(1 << CS20) ;

TCNT2 = 0; //очистить счетчик

OCR2A = 125 - 1; //8uS\*125=1mS (разрешить распространение тактовых импульсов)

TIMSK2 |= (1 << OCIE2A); //включить прерывание сравнения выходных данных

interrupts(); //включить прерывания

}

// ===============================

// loop()

// ===============================

void loop()

{

// ----- measure object distances

if (TaskFlag1)

{

TaskFlag1 = false;

measure();

// -----Distance1 and Distance2 readings to the display

Serial.print(Distance1); Serial.print(","); Serial.println(Distance2);

}

}

// ===============================

// task scheduler (1mS interrupt)

// ===============================

ISR(TIMER2\_COMPA\_vect)

{

// ----- timers

TaskTimer1++; //task 1 timer

// ----- task1

if (TaskTimer1 > 499) //интервал между пингами (50mS=423cm)

{

TaskTimer1 = 0; //таймер сброса

TaskFlag1 = true; //signal loop() to perform task

}

}

// ===============================

// measure distances

// ===============================

void measure()

{

// ----- locals

unsigned long start\_time; //микросекунды

unsigned long finish\_time1; //микросекунды

unsigned long finish\_time2; //микросекунды

unsigned long time\_taken; //микросекунды

boolean echo\_flag1; //флаги отражают состояние эхо-линии

boolean echo\_flag2;

// ----- send 10uS trigger pulse

digitalWrite(Trig1, HIGH);

digitalWrite(Trig2, HIGH);

delayMicroseconds(10);

digitalWrite(Trig1, LOW);

digitalWrite(Trig2, LOW);

// ----- wait for both echo lines to go high

while (!digitalRead(Echo1));

while (!digitalRead(Echo2));

// ----- record start time

start\_time = micros();

// ----- reset the flags

echo\_flag1 = false;

echo\_flag2 = false;

// ----- measure echo times

while ((!echo\_flag1) || (!echo\_flag2))

{

// ----- Echo1

if ((!echo\_flag1) && (!digitalRead(Echo1))) //получено Echo1

{

echo\_flag1 = true;

finish\_time1 = micros();

time\_taken = finish\_time1 - start\_time;

Distance1 = ((float)time\_taken) / 59; //использование 59, так как есть обратный путь

}

// ----- Echo2

if ((!echo\_flag2) && (!digitalRead(Echo2))) //получено Echo2

{

echo\_flag2 = true;

finish\_time2 = micros();

time\_taken = finish\_time2 - start\_time;

Distance2 = ((float)time\_taken) / 29.5; //использование 29.5, так как обратного пути нет

}

}

}

**Приложение Б**

Ниже представлен листинг кода, написанного в Processing IDE на Java:

Листинг Б.1:

// ----- последовательный порт

import processing.serial.\*; //Импорт последовательной библиотеки

Serial myPort; //Объект последовательного порта

final int Baud\_rate = 115200; //Скорость связи

String Input\_string; //Используется для входящих данных

// ----- пользовательская фигура

PShape Object; //Дать фигуре имя

int Frame\_count = 0; //Счетчик кадров

boolean Frame\_visible = true; //true=visible; false=invisible

// ----- отображение графики

PGraphics Canvas; //Название создаваемой области рисования

PFont myFont; //Название создаваемого шрифта

float Baseline = 100; //базовая линия треугольника (cm)

float X; //X координата в (cm)

float Y; //Y координата в (cm)

// ----- расстояние датчика НИЖЕ базовой линии (cm)

float Offset = 50; //Предполагает квадратный дисплей

// ----- файл

PrintWriter output;

// =========================

// установка

// ==========================

void setup() {

// ----- экран настройки

size(800, 800, P3D); //Определение размера окна, 3D

background(0); //Черный

frameRate(60); //60 кадров в секунду

// ----- создание области рисования для затухания луча

Canvas = createGraphics(width, height);

// ------ создание экранного шрифта

myFont = createFont("Arial Black", 20);

// ----- настройка объект

Object = createShape(ELLIPSE, 0, 0, 30, 30); //Создание объекта

Object.setFill(color(255, 0, 0, 255)); //красный, непрозрачный

Object.setStroke(color(255, 0, 0, 255)); //красный, непрозрачный

// ----- инициализация последовательного порта

/\*

ВАЖНО:

Если на дисплее не отображается arduino, следует попробовать

изменить [номер], связанный с COM-портом.

Строка кода "printArray(Serial.list());" генерирует список

[номеров] в квадратных скобках. например: [0] "COM5"

[Номер] внутри квадратной скобки ДОЛЖЕН совпадать с [номером] в строке

кода "myPort = new Serial(this, Serial.list()[0], Baud\_rate);"

\*/

printArray(Serial.list()); //Вывод имеющихся COM-портов на экране

myPort = new Serial(this, Serial.list()[1], Baud\_rate);

myPort.bufferUntil('\n');

//файл

output = createWriter("positions.txt");

}

// ==========================

// рисование

// ==========================

void draw()

{

// ----- обновить экран

background(0); //Черный экран

textFont(myFont, 20); //Указание шрифта, который будет использоваться

draw\_grid(); //Рисование сетки

draw\_object();

// ----- файл

point(X, Y);

output.println("Coordinate X: " + X + " Coordinate Y: " + Y);

}

// =======================

// последовательное событие (вызывается с каждой строкой данных Arduino)

// =======================

void serialEvent(Serial myPort)

{

// ----- дождитесь перевода строки

Input\_string = myPort.readStringUntil('\n');

println(Input\_string); //Визуальная обратная связь

// ----- утверждение

if (Input\_string != null)

{

// ----- обрезать пробелы

Input\_string = trim(Input\_string);

String[] values = split(Input\_string, ',');

// ----- сбор переменных Герона

float a = float(values[1]) - float(values[0]); //d2 (vertex -> sensor B)

float b = float(values[0]); //d1 (vertex -> sensor A)

float c = Baseline; //Базовая линия

//float d = c\*1.414; //диагональ дисплея (квадратная)

float d = sqrt(150\*150 + 100\*100); //диагональ (дисплей + смещение)

float s = (a + b + c)/2; //Полупериметр

// ----- проверка расстояний

/\* устранение фиктивныч ошибок \*/

boolean distances\_valid = true;

if

(

(a < 0) || //d1 должно быть меньше, чем d2

(b > d) || //d1 вне зоны действия

(a > d) || //d2 вне зоны действия

((s - a) < 0) || //Эти значения должны быть положительными

((s - b) < 0) ||

((s - c) < 0)

)

{

distances\_valid=false;

X=1000; //Переместить мигающую точку за пределы экрана

Y=1000; //Переместить мигающую точку за пределы экрана

}

// ----- применение формулы Герона

if (distances\_valid)

{

float area = sqrt(s \* (s - a) \* (s - b) \* (s - c));

Y = area \* 2 / c;

X = sqrt(b \* b - Y \* Y);

// ----- отображение данных для действительных эхо-сигналов

print(" d1: ");

println(b);

print(" d2: ");

println(a);

print(" base: ");

println(c);

print("offset: ");

println(Offset);

print(" s: ");

println(s);

print(" area: ");

println(area);

print(" X: ");

println(X);

print(" Y: ");

println(Y);

println("");

}

myPort.clear(); //Очистка буфера приема

}

}

// ==========================

// рисование сетки

// ==========================

void draw\_grid()

{

pushMatrix();

scale(0.8);

translate(width\*0.1, height\*0.10);

fill(0);

stroke(255);

// ----- граница

strokeWeight(4);

rect(0, 0, width, height, 20, 20, 20, 20);

// ----- горизонтальные линии

strokeWeight(1);

line(0, height\*0.1, width, height\*0.1);

line(0, height\*0.2, width, height\*0.2);

line(0, height\*0.3, width, height\*0.3);

line(0, height\*0.4, width, height\*0.4);

line(0, height\*0.5, width, height\*0.5);

line(0, height\*0.6, width, height\*0.6);

line(0, height\*0.7, width, height\*0.7);

line(0, height\*0.8, width, height\*0.8);

line(0, height\*0.9, width, height\*0.9);

// ----- вертикальные линии

line(width\*0.1, 0, width\*0.1, height);

line(width\*0.2, 0, width\*0.2, height);

line(width\*0.3, 0, width\*0.3, height);

line(width\*0.4, 0, width\*0.4, height);

line(width\*0.5, 0, width\*0.5, height);

line(width\*0.6, 0, width\*0.6, height);

line(width\*0.7, 0, width\*0.7, height);

line(width\*0.8, 0, width\*0.8, height);

line(width\*0.9, 0, width\*0.9, height);

// ----- обозначение оси X

fill(255); //Белый текст

textAlign(LEFT, TOP);

text("0", -20, height+10); //0cm

text("50", width\*0.5-20, height+10); //50cm

text("100cm", width-20, height+10); //100cm

// ----- обозначение оси Y

textAlign(RIGHT, BOTTOM);

text("100cm", 10, -10); //100cm

textAlign(RIGHT, CENTER);

text("50", -10, height/2); //100cm

popMatrix();

}

// ==========================

// рисование объекта

// ==========================

void draw\_object()

{

pushMatrix();

scale(0.8);

stroke(0, 255, 0);

strokeWeight(1);

translate(width\*0.1, height\*1.1); //(0,0) теперь нижний левый угол

// ----- сделать так, чтобы объект мигал

if ((frameCount-Frame\_count)>4)

{

Frame\_visible = !Frame\_visible;

Frame\_count = frameCount;

}

// ----- цветовая схема объекта

if (Frame\_visible)

{

// ----- сделать объект видимым

Object.setFill(color(255, 0, 0, 255)); //непрозрачный

Object.setStroke(color(255, 0, 0, 255)); //непрозрачный

} else

{

// ----- скрыть объект

Object.setFill(color(255, 0, 0, 0)); //Прозрачный

Object.setStroke(color(255, 0, 0, 0)); //Прозрачный

}

// ----- нарисуйте объект

pushMatrix();

translate(X/100\*width, -(Y-Offset)/100\*height);

shape(Object);

popMatrix();

popMatrix();

}

// ----- файл

void keyPressed() {

output.flush(); // записываем в файл оставшиеся данные

output.close(); // закрываем файл

exit(); // останавливаем программу

}

**Приложение В**

В приложении В расписан код для графического интерфейса, реализованного на языке программирования C# в Visual Studio 2019 IDE.

Листинг В.1:

using MaterialSkin.Controls;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Diagnostics;

namespace KotsichPractice

{

public partial class Form1 : MaterialForm

{

readonly MaterialSkin.MaterialSkinManager materialSkinManager;

public Form1()

{

InitializeComponent();

materialSkinManager = MaterialSkin.MaterialSkinManager.Instance;

materialSkinManager.EnforceBackcolorOnAllComponents = true;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkin.MaterialSkinManager.Themes.LIGHT;

materialSkinManager.ColorScheme = new MaterialSkin.ColorScheme(MaterialSkin.Primary.Indigo500, MaterialSkin.Primary.Indigo700, MaterialSkin.Primary.Indigo100, MaterialSkin.Accent.Pink200, MaterialSkin.TextShade.WHITE);

MaximizeBox = false;

}

private void materialLabel1\_Click(object sender, EventArgs e)

{

}

private void materialLabel2\_Click(object sender, EventArgs e)

{

}

private void tabPage1\_Click(object sender, EventArgs e)

{

}

private void materialButton1\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("https://github.com/Solitarrr/KotsichPractice/tree/master");

}

private void materialButton2\_Click(object sender, EventArgs e)

{

Process.Start(new ProcessStartInfo { FileName = @"..\..\..\ArduinoCodePractice\ArduinoCodePractice.ino", UseShellExecute = true });

}

private void materialButton3\_Click(object sender, EventArgs e)

{

Process.Start(new ProcessStartInfo { FileName = @"..\..\..\ProcessingCodePractice\ProcessingCodePractice.pde", UseShellExecute = true });

}

private void materialButton4\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

string FilePath = openFileDialog1.FileName;

Process.Start(FilePath);

}

}

private void materialButton5\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("https://www.tinkercad.com/things/9kK56aoCDLc?sharecode=EnIYgLRdEbR-k2v-Y98Zf5-2OuDiHo5eDEhO1v6m9e0");

}

private void materialButton6\_Click(object sender, EventArgs e)

{

if (pictureBox1.Visible == true)

{

pictureBox2.Visible = true;

pictureBox1.Visible = false;

}

else if (pictureBox2.Visible == true)

{

pictureBox1.Visible = true;

pictureBox2.Visible = false;

}

}

private void materialButton7\_Click(object sender, EventArgs e)

{

Process.Start(new ProcessStartInfo { FileName = @"..\..\..\ProcessingCodePractice\positions.txt", UseShellExecute = true });

}

private void materialButton8\_Click(object sender, EventArgs e)

{

System.Diagnostics.Process.Start("https://www.arduino.cc/en/software");

System.Diagnostics.Process.Start("https://processing.org/download");

}

private void materialLabel4\_Click(object sender, EventArgs e)

{

}

private void materialButton10\_Click(object sender, EventArgs e)

{

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}

**Приложение Г**

Последним приложением является приложение Г, содержащее в себе сканы бумаг, на которых изначально были нарисованы схемы, представлены расчеты и формулы, необходимые для реализации проекта.

На рисунке Г.1 представлена схема, показывающая как именно ультразвуковые датчики охватывают зону плоскости, а в правой части рисунка располагается схема треугольника с датчиками и формулами.

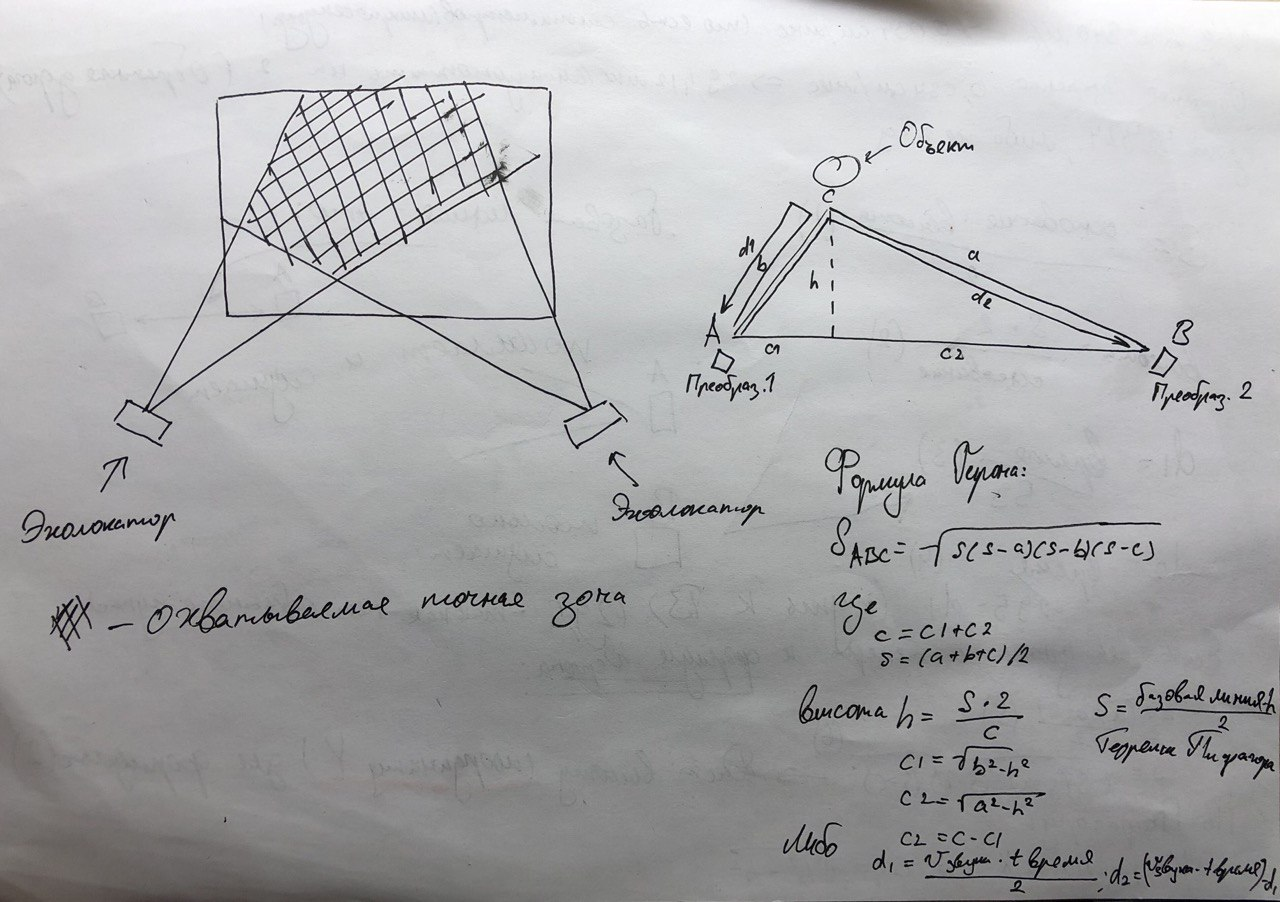


Рисунок Г.1 – Первая страница

На рисунке Г.2, расположенном ниже, представлены расчеты, которые были проведены перед тем, как работа была выполнена в программном и графическом вариантах, то есть на самом персональном компьютере.

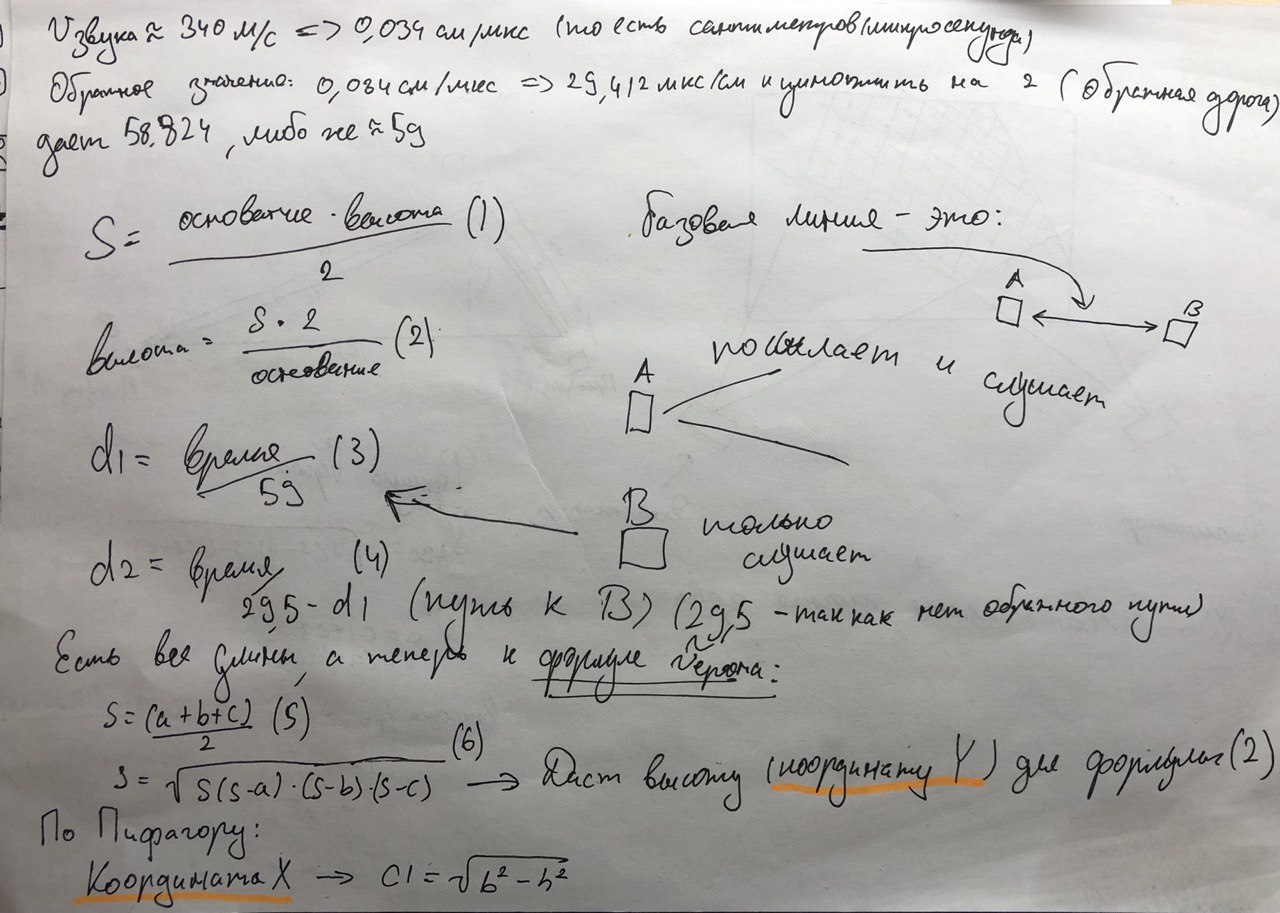


Рисунок Г.2 – Вторая страница

На рисунке Г.3 представлен вариант графического интерфейса и разделение вкладок для него. Данный вариант в конечном счете был реализован в Visual Studio 2019 IDE на таком языке программирования, как C#.

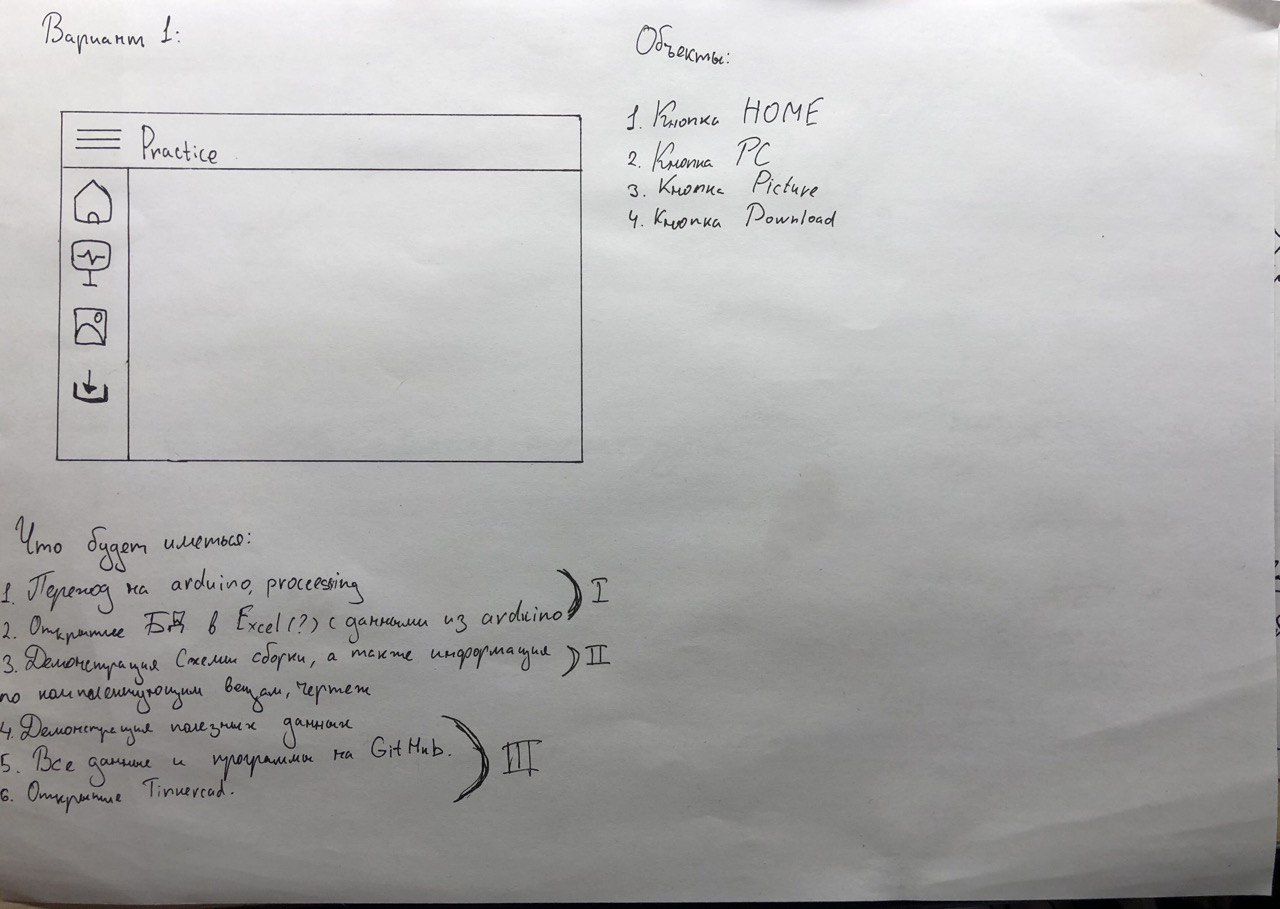


Рисунок Г.3 – Третья страница